Санкт-Петербургский государственный университет

Герасимов Владислав Юрьевич

Выпускная квалификационная работа

Сегментация математических выражений

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальные информатика и информационные технологии»
ООП СВ.5003.2016 «Программирование и информационные технологии»
Профиль «Автоматизация научных исследований»

Научный руководитель:
доцент, кафедра компьютерных технологий
и систем, к.ф.-м.н, Погожев С. В.

Санкт-Петербург
2020 г.

Содержание

Введение . . . 3

Обзор существующих решений . . . 4

Постановка задачи . . . 4

Обзор литературы . . . 5

Глава 1. Общий подход к решению . . . 7

Глава 2. Описание архитектуры . . . 8

2.1. Свёрточная нейронная сеть . . . 8

2.2. Рекуррентная нейронная сеть . . . 10

2.3. Encoder-decoder архитектура . . . 14

2.4. Механизм Attention . . . 15

Глава 3. Обзор моделей . . . 16

3.1. WYGIWYS . . . 16

3.2. SAT . . . 17

3.3. Im2Latex . . . 17

Глава 4. Данные . . . 19

Глава 5. Обучение и результаты . . . 22

Глава 6. Программная реализация . . . 25

Заключение . . . 26

Список литературы . . . 27

Введение

Задача распознавания символов (англ. optical character recognition, OCR) хорошо изучена и является очень важной составляющей любого проекта, который занимается автоматизацией документооборота или внедрения безбумажных технологий.

Распознавание применяется для преобразования документов различных форматов в текстовое представление. Такое представление позволяет осуществлять редактирование, поиск выражений, слов или фраз, хранить документ в более компактной форме, демонстрировать или распечатывать материал, не теряя качества, анализировать информацию, а также применять к тексту форматирование или преобразовывать текст в речь.

Отдельной важной подзадачей символьного распознавания документов является распознавание математических формул. Её важность и актуальность обусловлена тем, что большое количество уже существующих научных заметок, статей и книг содержат сложные математические конструкции, формулы и выражения, а доступ к текстовым вариантам таких изданий по тем или иным причинам оказывается затруднен.

Процесс распознавания математических выражений является более сложным процессом, нежели распознавание текста. Текст является одномерной последовательностью символов, математическая формула — двумерная конструкция. Текст может содержать только буквы или цифры, в математической формуле количество допустимых к использованию символов неограничено. В тексте отдельно взятые символы не имеют смысла (за исключением акронимов), а семантической единицей является слово, т.е. последовательный набор символов. В математической формуле каждый символ имеет свой смысл. Более того, один и тот же символ в разных контекстах может иметь разный смысл [1], [2].

В случае с математическими выражениями помимо распознавания непосредственно математического символа необходимо также уметь распознавать структуру всей формулы, а разбор неточностей осложняется тем фактом, что в языке математики отсутствует единый, универсальный словарь.

Исходя из перечисленных выше отличий, можно прийти к выводу о том, что задачи распознавания текстов и задачи распознавания математических формул должны иметь несколько разные подходы к решению.

Обзор существующих решений

Проанализируем уже существующие программные решения.

ABBYY FineReader [3] – коммерческое программное обеспечение с закрытым исходным кодом. Функционал доступен через приложение для настольных компьютеров. Поддерживает высокую точность распознавания текстов на большом количестве языков. Распознает только простые математические формулы.

Mathrix [4] – коммерческое программное обеспечение с закрытым исходным кодом. Доступно как через приложение для мобильных телефонов и компьютеров, так и через публичный API [5]. Обеспечивает высокую точность распознавания отдельных математических формул с английским текстом [6].

InftyReader [7] – коммерческое программное обеспечение с закрытым исходным кодом. Функционал доступен только через приложение для Windows. Обеспечивает неплохую точность распознавания страниц с текстом и математическими формулами. Имеется поддержка ряда языков, в том числе и русского [8].

Docs Matter [9] – коммерческое программное обеспечение с закрытым исходным кодом. Функционал доступен через мобильное приложение или SDK. Не распознает математические формулы.

Tesseract [10] – свободное программное обеспечение с открытым исходным кодом. Собирается под Linux, Windows, macOS. С помощью дополнительных модулей распознает простые математические формулы.

Среди всех описанных выше решений нет идеального, распознающего математические выражения со 100% точностью.

Постановка задачи

Можно поделить задачу распознавания математических выражений на следующие условные этапы:

1. Разбиение документа на текстовые, математические блоки, блоки с изображениями.

2. Сегментация отдельных формул, выделение структуры формулы.

3. Распознавание отдельных символов в формуле.

4. Перевод получившихся на предыдущем этапе структуры и символов в формат LaTeX.

5. Вывод результата.

Целью данной работы является разработка системы, способной выполнить этапы 2–5. Реализация первого этапа выходит за рамки данной работы: предполагается, что все исходные изображения априори содержат только одну печатную формулу без текстовых блоков.

Данная работа построена следующим образом. В Главе 1 описывается общий подход к решению задачи. В Главе 2 содержится описание нескольких общих архитектурных элементов. В Главе 3 представляются несколько архитектур глубоких сетей, которые могут быть использованы для решения задачи. Глава 4 описывает этап подготовки и генерации данных, а также их предобработку. В Главе 5 раскрываются некоторые детали обучения и анализируются результаты. В Главе 6 описывается разработанная программа.

Обзор литературы

Подойти к решению задачи можно разными путями. Один из них – решать задачу последовательно, ведь для решения каждого отдельно взятого шага, упомянутого ранее, уже существует свой набор алгоритмов.

Например, обзор нескольких популярных алгоритмов для этапа разбиения документа (этап №1) приведен в [15]. Методы решения этапа №2 приведены в [16]: их суть заключается в многоэтапном процессе построения дерева пространственных и семантических отношений между компонентами формулы. При наличии такой информации этап №4 (перевод структуры

в LaTeX формат) не составляет никакого труда. Этап №3 можно успешно решить, например, применением [17] сверточной нейронной сети (подробнее про сверточные сети – см. Глава 2.1).

Преимуществом такого подхода является предсказуемость поведения построенной системы. В то же время, поскольку система работает в несколько этапов, происходит постепенное накопление ошибки работы. Кроме того, некоторые детерминированные алгоритмы сегментации требуют наперед заданного набора четких правил семантических отношений символов [18].

Тем не менее в данной работе был использован другой, end-to-end подход, формально описанный в Главе 1. В этом подходе задача не делится явно на отдельные этапы, а полностью (этапы 2–4) решается с помощью глубокой нейронной сети с особой encoder-decoder архитектурой (подробное описание дано в Главе 2). Подход применяется в следующих работах: [28] – нейросеть для генерации подписи к изображению (отвечает на вопрос "Что происходит на изображении?"), [30], [31] – решения задачи Visual Q&A (по картинке ответить на вопрос произвольной сложности, например, какого цвета предмет), [35] – задача машинного перевода. Наконец, ключевыми для этой работы являются следующие статьи: в [32] дается описание encoder-decoder архитектуры, [27] представляет механизм Attention, а [28], [33], [34] описывают конкретные модели для решения image-to-latex задачи. Также полезными для ознакомления являются статьи об архитектурах CNN [17] и LSTM [24].

Глава 1. Общий подход к решению

Сформулируем задачу в следующем виде.

Пусть $x \in X$ – черно-белое изображение с шириной W и высотой H (т. е. $X \in R^{W \times H}$). Пусть также $y \in Y$ – последовательность $y_1, y_2, ..., y_P$, $y_i \in \Omega$, где Ω – некоторый словарь символов. Отображение $f : Y \to X$ – преобразование текстовой формулы в изображение. Задача состоит в нахождении приближенного обратного отображения $\tilde{f} \approx f^{-1}$.

Следует отметить, что отображение f (в случае с форматом LaTeX) не является взаимно-однозначным, а является сюръекцией: одному и тому же изображению может соответствовать несколько разных последовательностей. Строго говоря, в этом случае следует говорить о нахождении только правого обратного отображения.

Новую формулировку задачи можно легко свести к задаче машинного обучения с учителем. Входными данными в этом случае выступают пары (x, y) – картинка и формула. Функционал качества рассчитывается следующим образом. На выходе система выдает формулу $\tilde{y}$. С помощью средств визуализации из формулы генерируется изображение $\tilde{x}$, которое затем сравнивается с эталоном x. Величина, обратная расстоянию между $\tilde{x}$ и x, и есть степень качества распознавания.

Таким образом, мы показали, что теоретически решить задачу распознавания можно с использованием машинного обучения end-to-end, без явного разделения на этапы. Перейдем к описанию нескольких архитектур нейронных сетей.

Глава 2. Описание архитектуры

У всех моделей глубокого обучения, которые будут рассмотрены в следующей главе, имеется схожая структура и некоторые общие элементы, которые часто используются в областях распознавания образов и обработки естественного языка. Ознакомимся с этими элементами поподробнее.

2.1 Свёрточная нейронная сеть

Сверточная нейронная сеть (CNN) – специальная архитектура глубоких нейронных сетей, нацеленная на эффективное распознавание образов. Архитектура может быть успешно применена в задачах OCR [17]. Примеры сверточных сетей для распознавания неструктурированного текста приведены в [19], [20].

Принцип работы сверточных нейронных сетей основан на операции двумерной свертки. Сначала выбирается некоторая числовая матрица, называемая ядром. Далее ядро "скользит" над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель. Ядро повторяет эту процедуру с каждой локацией, над которой оно "скользит", преобразуя двумерную матрицу в другую все еще двумерную матрицу.

Сверточные нейронные сети строятся из 2 основных видов слоев: сверточные (convolutional) и субдискретизирующие (pooling).

Сверточный слой – это основной блок свёрточной нейронной сети. Представляет из себя применение операции свертки к выходам с предыдущего слоя. Получившиеся в результате матрицы называется картами признаков. Количество выходных карт определено до начала обучения и фиксировано. Веса ядер свертки являются обучаемыми параметрами. В целом, количество обучаемых параметров одного сверточного слоя можно посчитать, используя простую формулу: $P = I * O * W * H$, где I – количество входных карт, O – количество выходных карт, а W, H – размерность ядер (которая также опредена и фиксирована).

Помимо размера ядра у сверточного слоя есть еще 2 необучаемых па-

раметра – сдвиг (stride) и дополнение (padding) [22]. Сдвиг – это величина шага "скольжения"ядра вдоль осей изображения. Дополнение – количество нулевых пикселей, которые добавляются к горизонтальным и вертикальным краям изображения.

Субдискретизирующий слой призван снижать размерность получившихся с предыдущего слоя карт признаков. Каждое входное изображение делится на блоки фиксированного размера (заданного при построении модели) и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума (т. н. max pooling) – из блока берется пиксел с максимальным значением, все остальные пиксели отбрасываются. Преимущества max pooling перед другими функциями описаны в статье [21]. На выходе слоя получаются карты признаков меньшей размерности, что положительно влияет на скорость обучения. В этом слое нет обучаемых параметров.

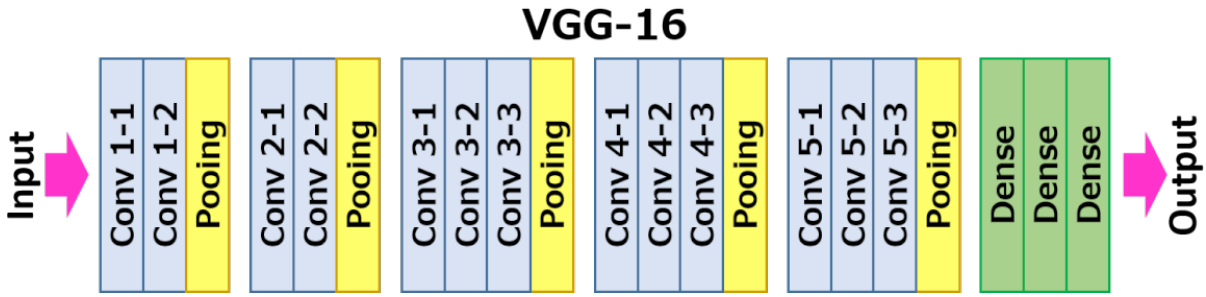


Рис. 1: Пример схемы сверточной сети. Здесь в качестве выходного слоя используется трехслойная полносвязная (Dense) нейронная сеть [42].

В задачах классификации в качестве выходного слоя CNN обычно используется простая полносвязная нейронная сеть с N – количеством выходных нейронов, равным количеству классов. Однако рассматриваемая задача является не задачей классификации, а задачей seq2seq – перевод одной последовательности в другую. Для работы с последовательностями элементов хорошо подходит другой вид нейронных сетей – рекуррентные нейронные сети.

2.2 Рекуррентная нейронная сеть

Рекуррентная нейронная сеть (RNN) – вид нейронных сетей, сети, содержащие обратные связи между элементами и позволяющие сохранять информацию. Нейрон в RNN вдобавок к новой порции входящих данных также получает некоторую информацию о предыдущем состоянии сети. Таким образом в таких сетях реализуется «память», что принципиально меняет характер ее работы и позволяет анализировать любые последовательности данных, в которых важно, в каком порядке идут значения.

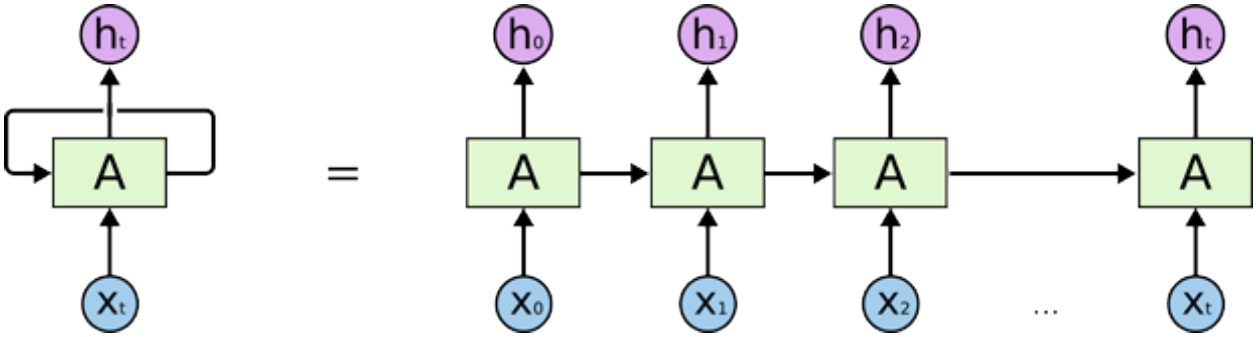


Рис. 2: Схема однослойной рекуррентной нейронной сети: на каждом этапе работы внутренний слой нейронов получает набор входных данных X и информацию о предыдущем состоянии внутреннего слоя A, на основании чего генерирует ответ h. Разница в архитектурах RNN кроется во внутреннем устройстве A [24].

Существует много различных архитектурных решений для рекуррентных сетей, но наибольшее распространение в последнее время получили сеть с долговременной и кратковременной памятью (LSTM) и управляемый рекуррентный блок (GRU).

Сеть с долговременной и кратковременной памятью (LSTM) – разновидность архитектуры рекуррентных нейронных сетей, устойчивая к проблеме долгосрочной памяти – ситуации, когда нейроны не имеют возможности надолго сохранить в памяти что-то, что обработали много циклов назад, какой бы важной та информация ни была.

Традиционный модуль LSTM [23] содержит четыре слоя нейронной сети, называемые вентилями (gates), и эти слои взаимодействуют особенным образом.

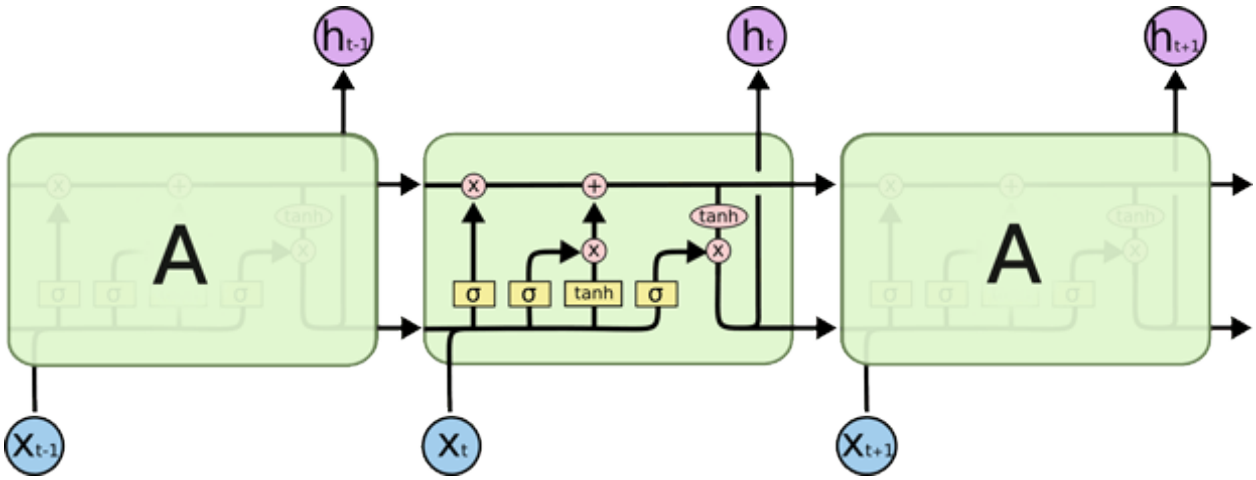
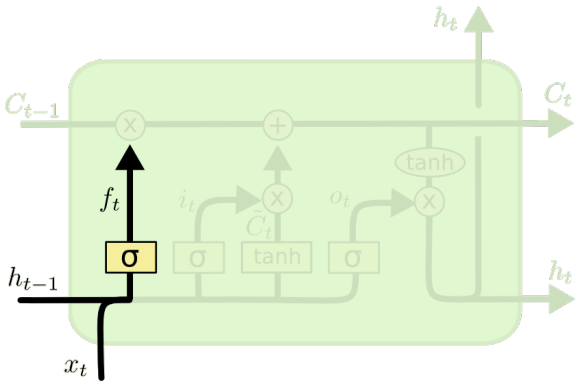


Рис. 3: Схема LSTM ячейки [24].

Ключевой компонент LSTM – состояние ячейки (горизонтальная линия, проходящая по верхней части схемы). Состояние ячейки проходит напрямую через всю цепочку, участвуя в нескольких линейных преобразованиях. С помощью линейных преобразований LSTM может удалять и записывать информацию в состояние ячейки; этот процесс регулируется вентилями.

«Вентиль забывания» определяет, какую информацию можно выбросить из состояния ячейки. Он принимает на вход h_{t-1} и x_t и возвращает число от 0 до 1 для каждого числа из состояния ячейки, где 1 означает «полностью сохранить», а 0 – «полностью выбросить». Функция активации в этом слое – сигмоида.

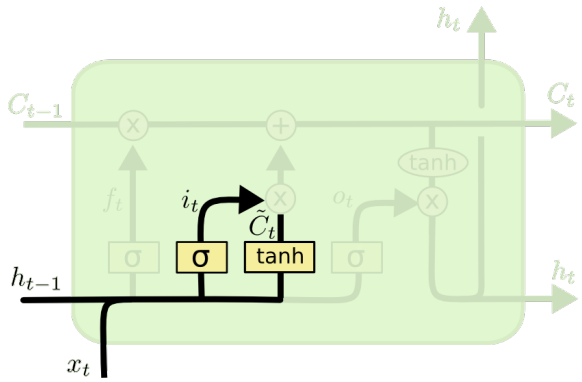


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рис. 4: Вентиль забывания [24].

Процесс записи новой информации разбит на две части. Сначала сигмоидальный слой («входной вентиль») определяет, какие значения следует

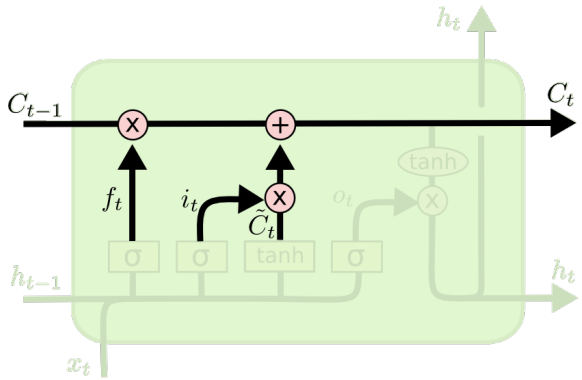
обновить. Затем tanh-слой строит вектор новых значений-кандидатов $\tilde{C}_t$, которые можно добавить в состояние ячейки.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рис. 5: Входной вентиль и tanh-слой [24].

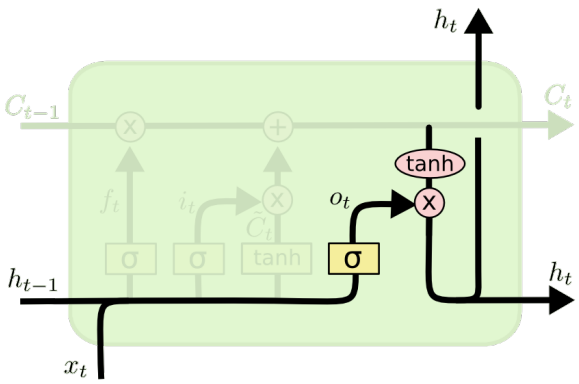
Замена старого состояния C_{t-1} на новое C_t производится по следующей формуле.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рис. 6: Обновление состояния [24].

Наконец, «выходной вентиль» отвечает за вывод информации из состояния ячейки. Он также является сигмоидальным слоем. Значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями вентиля, что позволяет выводить только требуемую информацию.



$$o_t = \sigma\left(W_o\ [\,h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рис. 7: Вывод информации. Выделенная сигмоида – выходной вентиль [24].

Описанная LSTM архитектура хорошо приспособлена к обучению в случаях, когда важные события разделены временными задержками с неопределённой продолжительностью и границами. У такой архитектуры есть и недостаток – большое количество обучаемых параметров (каждый «вентиль» является отдельным слоем). Уменьшить количество параметров призвана архитектура GRU.

Управляемые рекуррентные блоки (GRU) – архитектура рекуррентной нейронной сети, представленная в статье [25]. В отличие от LSTM, в этой архитектуре отсутствуют ячейка памяти C_t и выходной вентиль. При этом, было установлено [26], что ее эффективность при решении задач моделирования музыкальных и речевых сигналов сопоставима с использованием долгой краткосрочной памяти.

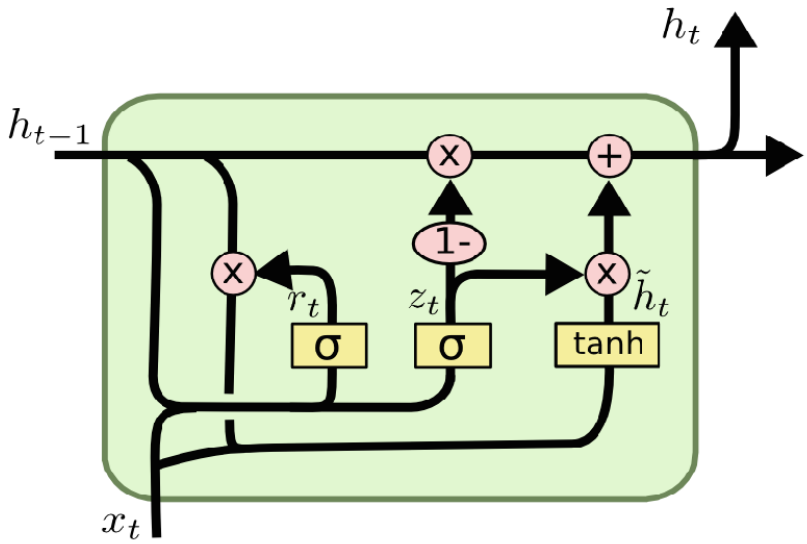


Рис. 8: Схема GRU ячейки [24].

2.3 Encoder-decoder архитектура

Как уже отмечалось ранее, описанные техники и виды нейронных сетей будут использованы как составные части одной глубокой сети. В настоящее время для решения задач seq2seq обработки естественных языков эффективно применяется архитектура Encoder-Decoder [32].

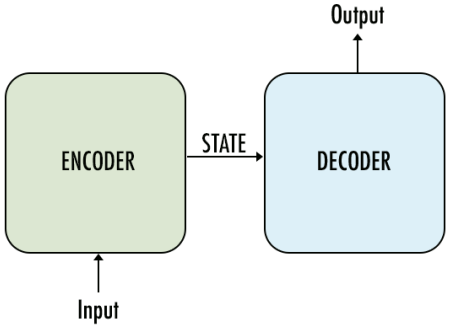


Рис. 9: Схема Encoder-Decoder архитектуры [41].

Архитектура включает в себя два компонента: кодировщик (Encoder) и декодировщик (Decoder). Кодировщик считывает всю входную последовательность и кодирует ее во внутреннее представление, часто это вектор фиксированной длины, называемый вектором контекста или состояния. Декодировщик в свою очередь считывает входную последовательность из кодировщика и генерирует выходную последовательность. Обе подмодели

обучаемы и обучаются одновременно.

В рассматриваемой задаче во всех моделях в качестве кодировщика будет использоваться CNN, а в качестве декодировщика – RNN. Между собой модели будут отличаться архитектурами архитектурами CNN, RNN и способе кодирования пространственной информации изображений.

2.4 Механизм Attention

Механизм Attention [27], [35], [28] применяется для работы с текстом, а также звуком и временными рядами. Используется как промежуточный этап Encoder-Decoder архитектуры. В основе лежит вычисление и использование корреляции между состояниями кодировщика и текущим состоянием декодировщика, которая дает декодировщику информацию о наиболее релевантных элементах последовательности (на какую часть картинки лучше «обратить внимание»). Процесс вычисления корреляции (т. е. назначения каждому элементу веса $0 \leq \alpha_t \leq 1$) также обучаем и выполняется с помощью полносвязной сети.

Глава 3. Обзор моделей

Для решения задачи были рассмотрены модели «WYGIWYS» [33], «Show, Attend and Tell» [28] (далее SAT) и «Im2Latex» [34]. Далее приведено их описание.

3.1 WYGIWYS

В модели WYGIWYS [33] в качестве кодировщика используется CNN с архитектурой, впервые описанной в [19].

Отличительной особенностью этой модели является наличие дополнительного промежуточного RNN, который порядово кодирует выходные из CNN карты признаков $V_{h,w}$. Начальное состояние для каждого ряда $\tilde{V}_{h,0}$ также обучаемо.

$$\tilde{V}_{h,w} = RNN(\tilde{V}_{h,w-1}, V_{h,w})$$

Таким образом в модели кодируется информация о вертикальном и горизонтальном расположении входящих в декодировщик элементов.

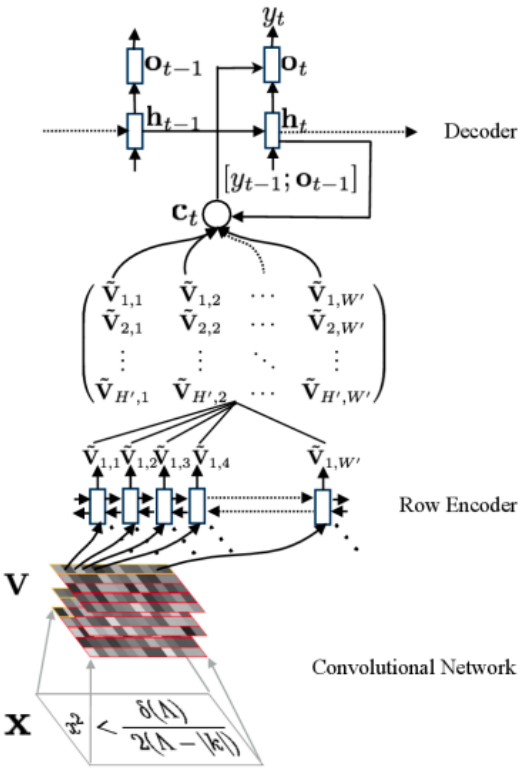


Рис. 10: Схема WYGIWYS из статьи [33].

В качестве декодировщика используется RNN с описанным ранее attention механизмом.

В общей сложности модель содержит около 9 млн. обучаемых параметров.

3.2 SAT

Модель SAT [28], изначально предназначенная для генерации текстового описания изображения (задача «image captioning»), также подходит и для текущей задачи: обе задачи сводятся к построению отображения $image \rightarrow text$, различие только в смысловой нагрузке полученного текста, а соответсвенно только в наборе данных для обучения.

Кодировщиком в этой модели является Oxford VGGnet [36]. Декодировщик в этой модели состоит из модифицированных (описанных в статье [37]) LSTM-слоев.

Используется модель soft attention – в качестве функции потерь используется дифференцируемая функция softmax, что позволяет модели обучаться через простой метод обратного распространения ошибки.

3.3 Im2Latex

Базовая архитектура в модели Im2Latex [34] остается такой же, как и в предыдущей модели. Однако здесь, также как и в первой модели, используется дополнительный этап пространственного кодирования карты признаков.

Процесс кодирования пространственной информации в этой модели следующий. Выходные из CNN векторы $a_{(h,w)} \in R^D$ карты признаков (размерность карты – $[H, W, D]$) конкатенируются с шагом $[S_H, S_W]$ (по вертикали и горизонтали). Получившийся набор векторов соответственно имеет размерность $[H/S_H, W/S_W, D * S_H * S_W]$, а каждый такой вектор соотносится с некоторой прямоугольной областью исходного изображения. Этот набор далее передается в декодировщик. В работе [35] было показано, что такой локальный пространственный анализ признаков на длинных формулах работает лучше, чем простое разворачивание карты признаков в один

вектор.

Глава 4. Данные

Для решения задачи был подготовлен датасет размером 150 тыс. изображений. Подготовка состояла из следующих этапов.

1. Выбор исходных данных.

   Все данные взяты из датасета соревнования KDD Cup 2003 [38]. Он содержит статьи в .tex формате по физике элементарных частиц с сайта arXiv.org, датированные вплоть до 2003 года.

2. Выбор отдельных формул из статей.

   Этот этап необходим, поскольку в исходной датасете представлены полные статьи с текстом, разделами и т. д. Использовались регулярные выражения для выявления следующих паттернов:

   - \begin{equation}... \end{equation}
   - $ ... $
   - \(... \)
   - \[... \]

   Остальные выражения игнорировались.

3. Нормализация формул.

   Нормализация включает в себя разбор и разделение выражений на токены (через пробел), отброс некоторых лишних токенов (комментарии, лишние экранирования, т. е. такие, которые не влияют на выходную картинку), приведение выражений к общему формату. Этот этап был выполнен с помощью библиотеки KaTeX. Стоит отметить, что библиотека не смогла обработать значительную часть формул, и все они были отброшены.

4. Генерация картинок из текста формул.

   Для всех формул была применена команда pdflatex из стандартного пакета TEXдля генерации pdf документа. При этом отбрасывались все

формулы, для которых инструмент не смог сгенерировать документ или количество страниц в документе больше одной.

Далее с помощью утилиты pdftoppm документы pdf были сконвертированы в формат png.

Параллельно с конвертаций, в отдельный файл записывались пары <название файла>–<номер формулы>.

5. Составление словаря.

   Так как формулы уже нормализованы, этот этап не составил никакого труда. В словарь были занесены все уникальные токены, за исключением тех, которые встретились во всех формулах всего один раз.

6. Фильтрация по количеству токенов.

7. Обрезка и downsampling.

   Сгенерированные на предыдущих этапах картинки имеют формат A4 с разрешением 300 dpi. При этом изображенные на них формулы занимают около 10% всей области. Для ускорения обучения все картинки были обрезаны по minimal bounding box формул. Также в 2 раза было понижено разрешение. На этом же этапе выявлялись пустые изображения путем проверки размеров bounding box. Такие изображения отбрасывались.

8. Padding.

   Были определены несколько допустимых размеров изображений. Каждое изображение было дополнено пустым белым пространством до ближайшего допустимого размера. Если изображение превышало максимальный размер, оно отбрасывалось.

В итоге получился датасет с 215 тыс. изображениями. Из-за технических ограничений было принято решение ограничить размер датасета до 150 тыс.: 120 тыс. картинок использовалось для обучения, 15 тыс. для теста и 15 тыс. для валидации. Итоговый словарь содержит около 400 токенов,

включая все буквы английского языка, цифры и математические обозначения. В словарь не вошел русский язык, поскольку в данных не было статей на русском языке. Максимальная длина формулы (количество токенов) – 150. Максимальный размер картинок – 500×160 пикселей.

Глава 5. Обучение и результаты

Для обучения были выбраны модели WYGIWYS и Im2Latex, поскольку Im2Latex и SAT принципиально отличаются только наличием пространственного кодирования в последней (которое также можно отключить).

Для оценки точности расспознавания на тестовой выборке были применены 3 метрики: расстояние Левенштейна, BLEU, точное совпадение.

Расстояние Левенштейна – метрика схожести двух строк, равная минимальному количеству операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

BLEU [39] – метрика схожести сгенерированной строки и эталона (в текущей задаче, исходной формулы). Метрика основана на сравнении всевозможных подпоследовательностей токенов длины N и подсчете количества совпадений. Подпоследовательность длины N носит название *N-граммa*. Расчет метрики производится по следующей формуле:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n log(p_n)),$$

где

$$w_n = 1/N$$

$$BP = \begin{cases} 1 & c > r \\ e^{1-r/c} & c \leq r \end{cases}$$

c – длина сгенерированной строки, r – длина эталона.

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{ngram \in C} Count_{clip}(ngram)}{\sum_{C' \in \{Candidates\}} \sum_{ngram' \in C'} Count(ngram')}$$

Здесь $Candidates$ – это весь сгенерированный корпус, функция $Count$ возвращает число вхождений N-граммы в сгенерированную строку, а функция

$Count_{clip}$ имеет следующий вид (аргумент убран для наглядности):

$$Count_{clip} = min(Count, MaxRefCount),$$

где $MaxRefCount$ есть максимальное число вхождений N-граммы во все эталоны (в рассматриваемой задаче эталон всегда один, поэтому эту функцию можно определить проще, как число вхождений N-граммы в эталон).

В формуле выше N обычно равняется 4.

Как было отмечено в Главе 1, одно и то же изображение в LaTeX можно представить через отличающиеся строки. Было бы несколько некорректно использовать для оценки качества распознавания только метрики близости строк, поэтому было также подсчитано точное совпадение. **Точное совпадение** – метрика, равная нормализованному числу сгенерированных изображений, попиксельно совпадающих с эталонным. Если из выходной формулы не удалось сгенерировать изображение, оно считается несовпадающим.

Модели обучались на одинаковых конфигурациях. Результаты представлены в следующей таблице. Аналогичные метрики были найдены для InftyReader.

	Левенштейна	BLEU	Точное совпадение
WYGIWYS (4 эпохи)	0.93	0.8649	0.749
Im2Latex (2 эпохи)	0.89	0.7800	0.626
InftyReader	0.55	0.6722	0.271

Из таблицы видно, что обе модели в целом значительно превзошли InftyReader, а модель WYGIWYS показала лучшие результаты. Меньше 0.1% сгенерированных моделями формул оказались синтаксически некорректными (не удалось создать для них изображение).

В скобках рядом с названием модели указано достигнутое количество эпох обучения (количество полных прохождений всех тренировочных данных). Поскольку модель Im2Latex значительно сложнее WYGIWYS и использует больше видеопамяти, потребуется больше времени, чтобы достигнуть аналогичных результатов.

Необходимо отметить, что в отличие от InftyReader (и вероятно других коммерческих продуктов), модели имеют узкую область применения – стандартный LaTeX шрифт, только английские символы, 300 dpi, без искажений и артефактов. Только в этом случае модели достигают результатов, отмеченных в таблице. Какие-либо отклонения в характеристиках данных резко снижают качество распознавания.

Глава 6. Программная реализация

Была разработана программа с UI, позволяющая обрабатывать изображения с использованием обеих моделей и выводить результат в .csv формат.

Исходный код программы доступен по ссылке [40], написан на Python 2.7 с использованием библиотек PyQt, matplotlib. Для работы также необходимы установленные инструменты Torch, CUDA 9.2, Tensorflow, Keras, Pandas.

В программе присутствует возможность изменить степень сжатия картинки при обработке (по умолчанию, размеры картинок уменьшаются в 2 раза), на случай если исходное разрешение отличается от рекомендованного.

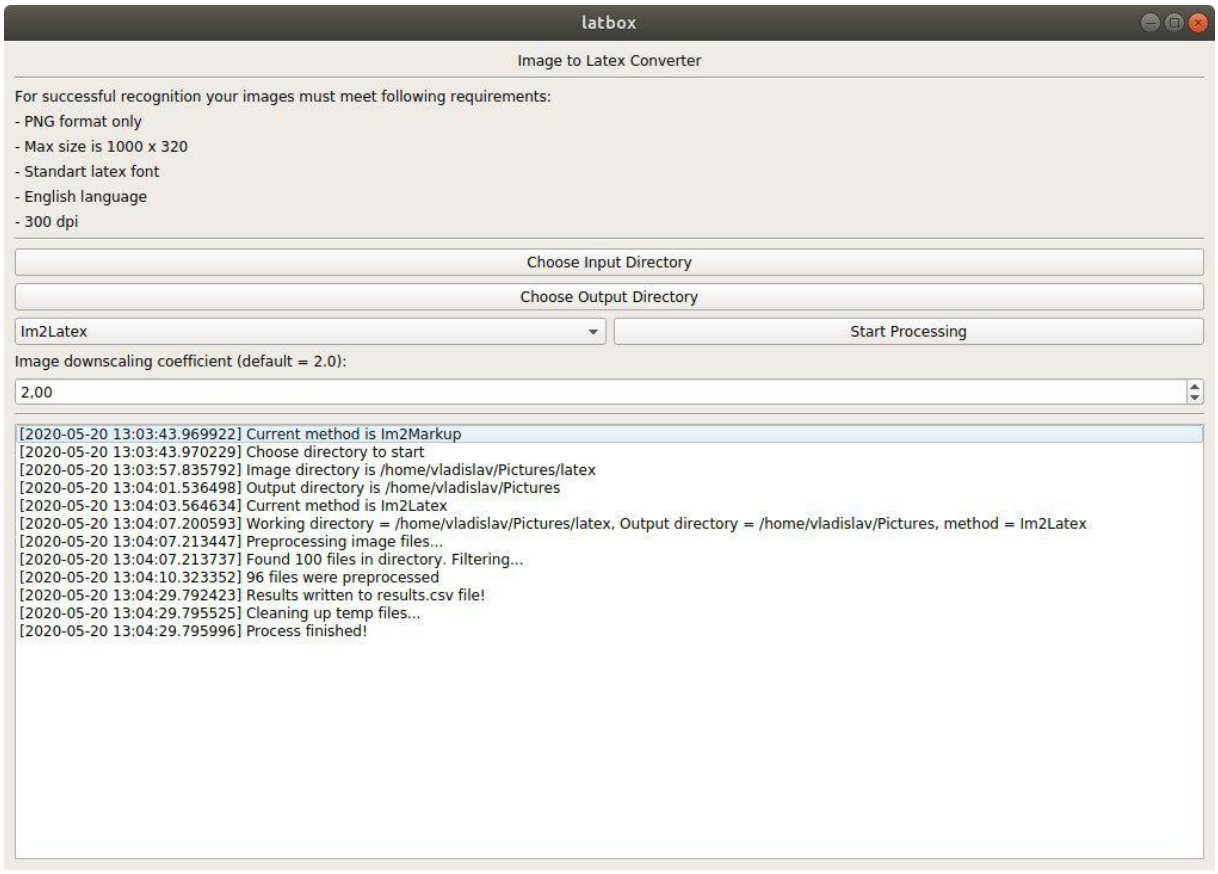


Рис. 11: Интерфейс программы.

Заключение

В работе изучена проблема распознавания математических выражений на изображениях. Были рассмотрены различные подходы для выделения и распознавания формул в тексте. В конечном счете был выбран end-to-end подход, использующий глубокие сети. Был подготовлен набор данных из пар корректных формул – изображений. Были рассмотрены различные модели глубоких сетей, две из них были обучены на подготовленных данных. Написана программа с пользовательским интерфейсом, позволяющая обработать входные изображения и выгрузить результат в .csv файл.

Модели показали очень хорошие результаты на подготовленном наборе данных. Тем не менее, как было отмечено в Главе 5, модели оказались крайне чувствительны к различным отклонениям характера данных: изменение разрешения, шрифта, добавление шумов резко снижает качество распознавания. Впрочем, такое поведение моделей нейронных сетей, обученных на однородных данных, ожидаемо, и если не выходить за установленные рамки, программой можно успешно пользоваться. В противном же случае система нуждается в доработке.

Список литературы

[1] Zanibbi R., Blostein D. Recognition and retrieval of mathematical expressions // Int. J. Doc. Anal. Recognit. 2012. Vol. 15, no. 4. P. 331–357. doi: 10.1007/s10032-011-0174-4.

[2] Jacob R. Bruce, Mathematical Expression Detection and Segmentation in Document Images. (2014).

[3] Сайт ABBYY FineReader. URL: https://www.abbyy.com/en-eu/finereader/ (дата обращения 14.03.2020).

[4] Сайт Mathpix. URL: https://mathpix.com/ (дата обращения 14.03.2020).

[5] Документация API Mathpix. URL: https://docs.mathpix.com/ (дата обращения 14.03.2020).

[6] Примеры результатов работы Mathrix. URL: https://github.com/Mathpix/api-examples/raw/master/results.pdf (дата обращения 14.03.2020).

[7] Сайт InftyReader. URL: http://www.inftyreader.org/ (дата обращения 15.03.2020).

[8] About InftyReader, ChattyInfty and InftyEditor. URL: http://www.inftyreader.org/?p=166 (дата обращения: 15.03.2020).

[9] Docs Matter. URL: https://www.yunmai.com/en/products.html (дата обращения 15.03.2020).

[10] Tesseract. URL: https://github.com/tesseract-ocr/ (дата обращения 15.03.2020).

[11] Theodore Bluche. Mathematical Formula Recognition using Machine Learning Techniques. (2010).

[12] Chaudhuri B.B. Digital Document Processing: Major Directions and Recent Advances (Advances in Pattern Recognition). Springer-Verlag, Berlin, Heidelberg, 2006.

[13] Garain U., Chaudhuri B.. On Development and Statistical Analysis of a Corpus for Printed and Handwritten Mathematical Expressions // Comput. Sci. Prepr. Arch. 2002. Vol. 2002, no. 7. P. 689–699. [Электронный ресурс]. URL: https://ssrn.com/abstract=3125396.

[14] Garain U., Chaudhuri B.B. A corpus for OCR research on mathematical expressions // Int. J. Doc. Anal. Recognit. 2005. Vol. 7, no. 4. P. 241–259. doi: 10.1007/s10032-004-0140-5.

[15] Shafait F., Keysers D., Breuel T.M. (2006) Performance Comparison of Six Algorithms for Page Segmentation. In: Bunke H., Spitz A.L. (eds) Document Analysis Systems VII. DAS 2006. Lecture Notes in Computer Science, vol 3872. Springer, Berlin, Heidelberg. doi: 10.1007/11669487_33

[16] Chan K.-F., Yeung D.-Y. Mathematical expression recognition: a survey // Int. J. Doc. Anal. Recognit. 2000. Vol. 3, no. 1. P. 3–15. doi: 10.1007/PL00013549.

[17] Lecun Y. et al. Gradient-based learning applied to document recognition // Proc. IEEE. 1998. Vol. 86, no. 11. P. 2278–2324. doi: 10.1109/5.726791.

[18] Alvaro F., S´nchez J.-A., Benedi J.-M. Recognition of Printed Mathematical Expressions Using Two-Dimensional Stochastic Context-Free Grammars // 2011 International Conference on Document Analysis and Recognition. IEEE, 2011. P. 1225–1229. doi: 10.1109/ICDAR.2011.247.

[19] Shi B., Bai X., Yao C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition // IEEE Trans. Pattern Anal. Mach. Intell. 2017. Vol. 39, no. 11. P. 2298–2304. doi: 10.1109/TPAMI.2016.2646371.

[20] Jaderberg M. et al. Deep Structured Output Learning for Unconstrained Text Recognition. 2014.

[21] Scherer D., Müller A., Behnke S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition BT – Artificial Neural

Networks – ICANN 2010 / ed. Diamantaras K., Duch W., Iliadis L.S. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 92–101.

[22] Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. 2016. [Электронный ресурс]. URL: http://arxiv.org/abs/1603.07285.

[23] Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Comput. 1997. Vol. 9, no. 8. P. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[24] Olah C. 2015. Understanding LSTM Networks. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs (дата обращения: 29.03.2020).

[25] Cho K. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014. [Электронный ресурс]. URL: http://arxiv.org/abs/1406.1078.

[26] Chung J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. 2014. [Электронный ресурс]. URL: http://arxiv.org/abs/1412.3555.

[27] Vaswani A. et al. Attention Is All You Need. 2017. [Электронный ресурс]. URL: http://arxiv.org/abs/1706.03762.

[28] Xu K. et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. 2015. [Электронный ресурс]. URL: http://arxiv.org/abs/1502.03044.

[29] Mnih V. et al. Recurrent Models of Visual Attention // Advances in Neural Information Processing Systems 27 / ed. Ghahramani Z. et al. Curran Associates, Inc., 2014. P. 2204–2212.

[30] Hermann K.M. et al. Teaching Machines to Read and Comprehend. 2015. [Электронный ресурс]. URL: http://arxiv.org/abs/1506.03340.

[31] Kazemi V., Elqursh A. Show, Ask, Attend, and Answer: A Strong Baseline For Visual Question Answering. 2017. [Электронный ресурс]. URL: http://arxiv.org/abs/1704.03162.

[32] Sutskever I., Vinyals O., Le Q. V. Sequence to Sequence Learning with Neural Networks. 2014. [Электронный ресурс]. URL: http://arxiv.org/abs/1409.3215.

[33] Deng Y. et al. Image-to-Markup Generation with Coarse-to-Fine Attention. 2016. [Электронный ресурс]. URL: http://arxiv.org/abs/1609.04938.

[34] Singh S.S. Teaching Machines to Code: Neural Markup Generation with Visual Attention. 2018. [Электронный ресурс]. URL: http://arxiv.org/abs/1802.05415.

[35] Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. [Электронный ресурс]. URL: http://arxiv.org/abs/1409.0473.

[36] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition // 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. 2015.

[37] Zaremba W., Sutskever I., Vinyals O. Recurrent Neural Network Regularization. 2014. [Электронный ресурс]. URL: http://arxiv.org/abs/1409.2329.

[38] KDD Cup 2003 datasets.
URL: http://www.cs.cornell.edu/projects/kddcup/datasets.html (дата обращения 10.04.2020).

[39] Papineni K. et al. IBM Research Report Bleu : a Method for Automatic Evaluation of Machine Translation // Science. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, 2001. Vol. 22176. P. 1–10. doi: 10.3115/1073083.1073135.

[40] Проект latbox. URL: https://bitbucket.org/voismager/latbox/

[41] Manish Chablani. 2017. Sequence to sequence model: Introduction and concepts. [Электронный ресурс]. URL:

https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d (дата обращения: 05.04.2020).

[42] Muneeb ul Hassan. 2018. VGG16 – Convolutional Network for Classification and Detection. [Электронный ресурс]. URL: https://neurohive.io/en/popular-networks/vgg16 (дата обращения: 28.03.2020).